REMARKS

In the Office Action, the Examiner indicated that claims 1 through 22 are pending in the application and the Examiner rejected all of the claims.

The Claim Objection

On page 2 of the Office Action, the Examiner objected to claim 1 for a minor informality. Applicant has amended claim 1 in accordance with the Examiner's suggestion. Accordingly, applicant respectfully requests that the Examiner reconsider and withdraw the objection to claim 1.

Rejection under 35 U.S.C. §103

On page 2 of the Office Action, the Examiner rejected claims 1-22 under 35 U.S.C. §103(a) as unpatentable over Raj Srinivasan ("RFC 1833: Binding Protocols for ONC PRC Version 2", hereinafter "Srinivasan") in view of Simson Garfinkel et al. ("Practical UNIX & Internet Security", hereinafter "Garfinkel") and further in view of Bill Venners ("Finding Services with the Jini Lookup Service," hereinafter "Venners").

The Present Invention

As exemplified by present independent claim 1, the present invention is a method of enabling a client, running on a first computing device that is connected to a second computing device, to use a service on that second computing device, comprising the steps of:

(a) a service, installed on the second computing device, registering its published name with a service broker on that second computing device;

(b) the client sending a message to the service broker specifying the name of the service;

wherein the published name of the service conforms to a structured naming convention that uniquely identifies the service as a service from a particular vendor, but without specifying the connection point address of that service, to enable the service broker to start up the service without the risk of a clash.

Services installed on a computing device register their published name, which conforms to a structured naming convention, such as reversed domain information, with a 'service broker' on that device. The service broker uses a single well-known port number address. When an external client, connected to the computing device that has a service broker, wants to use a service on that computing device, it sends a message to the service broker using the well known port number. The message specifies the name of the desired server and requests that the service broker inform it of the appropriate connection point (e.g. port number) to use. There is no dependency on port numbers or unstructured and arbitrary naming conventions.

The Examiner Has Not Established a Prima Facie Case of Obviousness

As set forth in the MPEP:

> To support a rejection under 35 U.S.C. §103, a reason, suggestion, or motivation to lead an inventor to combine two or more references must be found. *Pro-Mold and Tool Co. v. Great Lakes Plastics Inc.*, 37 U.S.P.Q.2d 1627, 1629 (Fed.Cir. 1996). The Examiner has not met his burden in establishing a reason, suggestion, or motivation for combining the cited references.

The Examiner incorrectly equates the claimed "published name" of the present invention to the "transport address of Srinivasan; however, the two are in fact very different. The published name of the service referred to in the independent claims of the present invention conforms to a structured naming convention, but specifically does not specify the connection point address of the service. The published name of the present invention is an identifier by which a client can specify a service when it does not have all the information necessary to communicate directly with the server. It is clear from the present specification this that the published name claimed in the present invention cannot include a port number relating to the service. For example, paragraph [0006] states that "The message specifies the name of the desired server and *requests that the service broker inform it* of the appropriate connection point (e.g., port number) to use". In other words, the published name must get the port number, if needed, from somewhere else.

Srinivasan, on the other hand, describes the "transport address" as follows:

> *The transport address, in turn, consists of a network address and a transport selector. In the case of a service available over TCP/IP or UDP/IP, the network address will be an IP address, and the transport selector will be a TCP or UDP port number.*
>
> (Srinivasan, page 1, final paragraph)

In the case of TCP/IP and UDP/IP protocols (as used in Srinivasan), the transport address described by Srinivasan necessarily *includes the service's port number* and therefore cannot be equivalent to the "published name" of our independent claims. There is no suggestion in Srinivasan that protocols other than TCP/IP and UDP/IP could be used, but it is clear from the quoted portion above that the transport address will always include the network address and transport selector and thus evidently includes the "connection point address" that claim 1 requires the published name must not to include.

Clearly, the "transport address" of Srinivasan does not teach or suggest the registration or specification of a "published name", as required by the independent claims.

The Examiner also asserts that Garfinkel describes automatically starting up the service when the request is received. In fact, however, the only reference to automatic start up is in bullet points at the top of page 2 of Garfinkel. These bullet points describe the things that must happen for an RPC client to communicate with an RPC server. Importantly, these are the general prerequisites for the communication to take place – they are not a description of the functionality of the techniques described in Garfinkel. The bullet points include the following: "The client and server must agree to communicate on a particular TCP or UDP port."

In the subsequent paragraphs, Garfinkel goes on to describe that "the simplest way to satisfy this list of conditions [the bullet points] is to have the UNIX computer start the server when the computer boots." In this embodiment, the server is clearly not automatically started when the request is received – it is instead started when its host server boots.

Garfinkel goes on to state the following:

> *When an RPC server starts, it dynamically obtains a free UDP or TCP port, **then** registers itself with the* portmapper. *When a client wishes to communicate with a particular server, it contacts the* portmapper *process, determines the port number used by the server, and then initiates communication.*
>
> (Garfinkel, page 2, emphasis added)

The claims of the present invention are directed to the situation where the client and server are linked through a service broker. If Srinivasan and Garfinkel really lie in an analogous field, they should therefore both concern service brokers. In fact, however, the only broker-like element that Garfinkel describes is the portmapper process, which provides the port number of a service to a client. However, Garfinkel's portmapper has nothing to do with the service until the

service registers itself with the portmapper (see quoted section go Garfinkel, above). Only once this registration takes place does the portmapper know the RPC service number and port number of the service, at which point in time it becomes able to facilitate interactions between the client (which knows the service number) and the service (whose port number is now known to the portmapper, but not to the client). It follows that it is impossible for the portmapper of Garfinkel to automatically start up the service (a function required of the broker in claim 1 of the present invention) until the service is registered with the portmapper – by which time the service must already be running.

Garfinkel does mention, in passing, the automatic starting up of a service, but provides no indication of how this automatic starting-up might be effected. However, what is perfectly clear from Garfinkel is that it cannot be the portmapper that starts up the service.

In view of the above, it is clear that neither of Srinivasan or Garfinkel , alone or in combination, teach or suggest the service by a service broker, as claimed in the present claims.

Finally, turning to Venners, nowhere in Venners is there a description of starting up of the service by the service broker, required by the present claims (and the Examiner does not suggest that this feature is present in Venners). Since this element is expressly found in each of the present claims, and since no combination of the three citations teaches or suggest this feature, all of the claims of the present invention are patentable over Srinivasan, Garfinkel, and/or Venners, taken alone or in combination.

Venners was cited by the Examiner due to its description of "reverse naming". However, the only parameter that is reverse named in Venners is the service type. The service type is by no means equivalent to its published name. Instead, it is quite literally the type of service that is offered (and is not necessarily even unique).

Venners discloses performing a look up for a Java service based on criteria expressed in a "template" object of the ServiceTemplate class discussed on page 2. Objects of this class include an array of serviceTypes (the type of service) and a serviced ("which uniquely identifies a service"). This is explained in the following passage from page 7:

> *Jini services are registered with a service ID that is globally unique and permanently maintained by each service, a service object, and any number of attribute objects. Using the ServiceTemplate, clients can look up a service by serviced, by the Java type or types of the service object, and by wildcard and exact matching of the attributes. [...] If you enter a LAN environment for the first time and you want to use a printer, you don't have to figure out the printer service's registered name; instead, you just look up the services that implement a well-known printer interface. Lookup by type also ensures that Jini clients will know how to use whatever is returned from their query, because they had to have knowledge of the type before sending their query.*
>
> (Venners, page 7, 2nd paragraph)

The types of service can be expressed using reverse naming. However, it is just the service types that are expressed in this way and not their published names. In the quote above, the distinction between a service type and its name is made very clear – the client can look up a printer service using its type when the client has no idea as to the name of the printer service (or even which particular printer services are available).

The serviceID, on the other hand, is used as a name to identify a particular service. However, there is no suggestion in Venners that the serviceID identifies the service as originating from a particular vendor. Instead, the serviceID is used to uniquely identify the service itself. This interpretation is supported by Sun's specification of the ServiceID class, a copy of which is attached to this response. The comments in Sun's specification make it clear that an object of the class serviced is a 128-bit unique identifier and describes its format and generation. There is no suggestion in Venners that the serviceID includes any indication of the service's vendor, and the fact that it does not is confirmed in Sun's specification.

Since neither the automatic starting up of a service by a service broker nor the use of a published name for a service that identifies the service's vendor is disclosed in any combination of the three citations, the subject matter of the claims is not obvious in view of these documents.

Accordingly, the Examiner is respectfully requested to reconsider and withdraw the rejection of claims 1-22 under 35 USC §103.

Conclusion

The present invention is not taught or suggested by the prior art. Accordingly, the Examiner is respectfully requested to reconsider and withdraw the rejection of the claims. An early Notice of Allowance is earnestly solicited.

The Commissioner is hereby authorized to charge any fees associated with this communication to applicant's Deposit Account No. 19-5425.

Respectfully submitted

February 19, 2008
Date

/Mark D. Simpson/
Mark D. Simpson, Esquire
Registration No. 32,942

Synnestvedt & Lechner LLP
112 Nassau Street
P.O. Box 592
Princeton, NJ 08542-0592